(4)

AFGL-TR-89-0290

Optical Instrumentation Support for the Airborne Ionospheric Observatory

Robert H. Eather
Cyril A. Lance

Keo Consultants
27 Irving St.,
Brookline, MA 02146

25 October, 1989

Final Report
12 March 1986 - 30 September 1989

Approved for public release; distribution unlimited

GEOPHYSICS LABORATORY
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
HANSCOM AIR FORCE BASE, MASSACHUSETTS 01731-5000

"This technical report has been reviewed and is approved for publication"

HOWARD W. KUENZLER
Contract Manager

WILLIAM K. VICKERY
Branch Chief

FOR THE COMMANDER

ROBERT A. SKRIVANEK
Division Director

This report has been reviewed by the ESD Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS).

Qualified requestors may obtain additional copies from Defense Technical Information Center. All others should apply to the National Technical Information Service.

If your address has changed, or if you wish to be removed from the mailing list, or if the addressee is no longer employed by your organization, please notify GL/IMA, Hanscom AFB, MA 01731. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notice on a specific document requires that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): KEO - FINAL	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFGL-TR-89- 0290
6a. NAME OF PERFORMING ORGANIZATION: KEO CONSULTANTS; 6b. OFFICE SYMBOL (If applicable):	7a. NAME OF MONITORING ORGANIZATION: GEOPHYSICS LABORATORY
6c. ADDRESS (City, State, and ZIP Code): 27 IRVING ST. BROOKLINE MA 02146	7b. ADDRESS (City, State, and ZIP Code): HANSCOM AFB MA 01731-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: GEOPHYSICS LAB.; 8b. OFFICE SYMBOL (If applicable): LIS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: F19628-86-C-0028
8c. ADDRESS (City, State, and ZIP Code): HANSCOM AFB MA 01731-5000	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO 62101F; PROJECT NO. 4643; TASK NO 08; WORK UNIT ACCESSION NO AK

11. TITLE (Include Security Classification)
OPTICAL INSTRUMENTATION SUPPORT FOR THE AIRBORNE IONOSPHERIC LABORATORY

12. PERSONAL AUTHOR(S)
ROBERT H. EATHER AND CYRIL A. LANCE

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
FINAL	FROM 3/12/86 TO 9/30/89	1989 OCT. 25	38

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
ALL-SKY IMAGING PHOTOMETER, PHOTOMETER, SPECTROPHOTOMETER, AURORA, AIRGLOW.

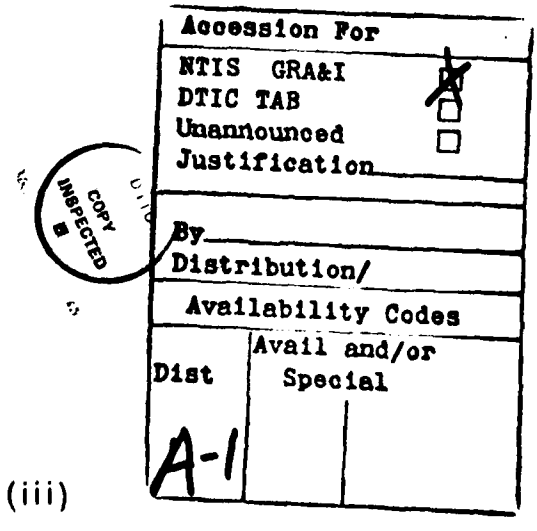
19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This Report summarizes activities during this 3½ year contract, with special emphasis on Year 3. Included is a description of software development for instrument control and calibration, hardware modifications and improvements, mission support for field trips, optics improvements, instrument maintennance, and data analysis.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Howard Kuenzler	22b. TELEPHONE (Include Area Code): 617 377 4789; 22c. OFFICE SYMBOL: LIS

DD Form 1473, JUN 86 *Previous editions are obsolete.*

Table of Contents



1. Contract Objectives:

To participate in ionospheric research programs using the new optical equipment (designed and constructed by Keo Consultants) that was planned to be installed on the Airborne Ionospheric Observatory (AIO), a research aircraft operated by the Ionospheric Effects Branch at AFGL. This participation to include:

(a) Testing, improvement and continuing development of the software operating systems that control the instrument operation.

(b) Complete testing, calibration and documentation of all optical properties of the optical systems.

(c) Improvements and continuing development of data recording systems for all instruments, and techniques for quick-look data presentation.

(d) Assistance in analysis of optical data obtained on airglow and auroral experiments.

(e) Provide personnel assistance on research flights and field trips as required by AFGL.

This Final Report summarizes efforts completed on this contract from March, 1986 to Sept.,1989. Much of this work has been described in previous reports (Scientific Report #1 and Scientific Report #2) which are referenced throughout this Final Report.

2. Changed Contract Objectives during Year I:

The optical equipment involved in this Contract was to be installed on the AIO. The photometers and spectrometers were sent to Wright Patterson AFB for this installation, and the equipment was held there for some nine months and so was unavailable at AFGL for checkout, software development and improvements. Consequently we did not hire a full-time software engineer on this Contract as early as planned, and continued to utilize a part-time engineer to continue development work on the software for the all-sky imaging photometer (ASIP II).

Towards the end of Year 1 it became evident that the optical equipment would probably not be installed permanently on the AIO. Consequently the emphasis of this Contract changed somewhat to address optimization of the use of ASIP II from the ground in association with various field trips of the Ionospheric Effects Branch, and the facilitation of image analysis. The net result of this change in direction was a

decreased level of spending on salaries and an increased level of spending on equipment. However the spending level by the end of Year 2 was still less than anticipated and allowed a no-cost extension for Keo providing the services listed under 1. above from April 15, 1989 to Sept. 30, 1989.

3.Software Development:

(a) Photometers and Spectrometers:

Year 1:

Much of the spectrophotometer software was rewritten to correct problems with the Spectrophotometer Controller operation. It also became apparent that the large number of interrupts that the computer had to generate to control the total of 10 stepper motors in the system was leading to timing problems so that only one spectrometer could be operated simultaneously with the photometer (not two as planned). A hardware solution utilizing intelligent stepper-motor control chips was devised but not implemented in Year 1 as the equipment was at Wright Patterson AFB and was not available to work on. (On the return of the equipment to AFGL in Year 2, it was decided to delay installation in the AIO to a later date) The recommended solutions to the problem described above were discussed and fully documented in the report entitled "Suggested Upgrades for the Photometer/Spectrometer System", and provided to AFGL. An abstracted version was included as Appendix I of Scientific Report #2 (AFGL-TR-88-0188).

Year 2:

No software development. for photometers or spectrometers.

Year 3:

In August of 1989, it was decided to use the photometer as a ground-based instrument for an upcoming campaign in Sondestrom, Greenland. Software was rewritten for the photometer system and debugged as a field instrument, including the hardware/software modification of providing an additional external port to track the position of the SRI radar. This external port was implemented in a generalized way that could be adapted to many field applications. During this work, the first generation software were debugged and tested thoroughly. Full testing of the

system will be continued after the first field operation at Sondestrom in October-November, 1989.

Four test programs were also written to facilitate trouble-shooting the system and checking various components. Data handling files were written to allow primitive data viewing capabilities in the field. A full set of command files were written to allow generalized construction of the software, and all software pneumonics were changed to make the code much more "user friendly". In addition, the software data file structures were re-written to allow a much more space efficient program, and thus give the operator much more flexibility in modifying the code for specified field needs. A full user's manual for this generation of software was written and included in the documentation set along with an updated software listing and updated hardware documentation. This manual contains an in-depth index of programmer notes to facilitate the easy modification of code to implement further field options (included here as Appendix I).

(b) ASIP II:

Year 1:

The imager software was completely exercised and tested, and in the process, some new routines were written and problems with original routines fixed. The complete package was then made more "user friendly" by introducing menu control of most operational functions. The menu system was designed for convenient and efficient operator interaction, but at the same time protecting the operating system from questionable or invalid commands.

The resultant software package was fully documented in the report entitled "All-Sky Imaging Photometer - Keo System Users Manual" which was included as Appendix I.in Scientific Report #1 (AFGL-TR-87-0224).

Year 2:

Software was developed for automatic calibration of ASIP II. The software takes the operator through a complete system calibration with appropriate screen instructions and prompts, with the full procedure taking about 40 minutes. This software was documented in the report entitled "ASIP II - All Sky Imaging Photometer - Calibration Software User's Manual" supplied to AFGL, and included as Appendix II in Scientific Report #2 (AFGL-TR-88-0188).

Further software improvements were made in Year 2 to upgrade ASIP's ability to keep accurate real (civil) time. (This improvement in system time was necessary for an Arecibo experiment where it was important to maintain synchronization with the on/off times of the HF Heater and the start/end times of ASIP exposures.) This Arecibo experiment also required new software to subtract the on/off images and to correct for system vignetting of the original ASIP II optics (since improved, see Section 4 below). The resultant images showed the capability of detecting airglow enhancements in 5577 OI and 6300 OI emissions of just 5 Rayleighs, with good contrast, and so demonstrated the powerful low-light-level capabilities of the instrument.

Near the end of Year 2, Keo began development of a new system to record and process scintillations on VHF/UHF satellite signals. The system is based on a Zenith Z-248 (IBM clone) computer and written in "C". Various commercial software packages in C were purchased to begin this development. Initial specifications of the development environment was completed in April, 1988; coding of the prototype system began in May, with prototype software testing in August, 1988. This effort suffered from the unexpected resignation of the software engineer (Mr. Keane). His replacement (Dr. Tsuei) initially worked with AFGL personnel (Dr. Sheehan) to continue the project (which was poorly documented at that stage), and the project was then taken over full time by Dr. Sheehan.

Year 3:

Software was written to allow recording of ASIP II images on the new Panasonic optical disc recorder. Image sequences through different filters can be recorded on different parts of the disc, so as to allow continuous playback of image sequences at particular wavelengths.

Keo personnel (Mr. Keane) also worked extensively on the new image analysis system delivered by Northwest Research. A large effort involved initial shakedown problems, understanding the system and its capabilities and limitations, and writing modified and new software for specific image analysis and presentation requirements. This system is now used regularly on a variety of tasks, and is proving extremely valuable to the various projects of the Ionospheric Effects Branch. Future upgrades have been suggested to reduce analysis time, and to allow compatibility with ASIP I images.

Dr. Tsuei also worked in improving various aspects of the Northwest Research software, including facilitating data interchange between the NW Research system and the ASIP II. In addition, he wrote a new utility so that an image pixel on the all-sky image can be individually addressed, and be associated with its corresponding azimuth and elevation coordinates. Selecting any two pixels allows an intensity profile along the line joining them to be displayed and written to a data tape format for distribution to other labs participating in auroral experiments. This has greatly facilitated the use of ASIPII as a quantitative instrument whose data can be shared amongst the auroral community

4.Optics Development:

Year 1:

(a) A new telescope spider was designed and fabricated to hold the diagonal mirror and front cover glass of the azimuth/elevation scanning telescope for the photometer system. This was necessitated because the original was broken when returned from Wright Patterson AFB. The new design is much more rugged.

(b) A diagonal viewing mirror/telescope was designed and fabricated to slide into the optics path of the ASIP II between the two relay lenses. This allows direct viewing of the output screen of the image intensifier (at selectable magnifications), and is very useful when focussing or trouble shooting.

(c) A new calibration light source was designed and fabricated for use in calibrating the ASIP II (and for all optical calibration procedures in general). It features a large output screen (100mm dia.) and uses integrating spheres to assure uniformity, and to allow a large dynamic range with aperture stops (with no consequent change in output spectral distribution). There is also provision for the insertion of an interference filter in a parallel-beam portion of the light path so as to obtain a monochromatic (30A) output screen. The available dynamic range exceeds 10**6. The light source is a tungsten halogen light with accurate current control. The unit was calibrated against a NBS secondary standard source, and has been delivered to AFGL.

(d) A set of five primary telecentric lenses for ASIP II was designed and fabricated. A different telecentric lens pair is used for each Mamiya primary lens (covering fields of view from 5 - 180 degrees). The optics (resolution, vignetting) was considerably improved by replacing the original telecentric element (a bi-convex

lens) with the plano-convex lens pair. The complete lens set (in a custom carrying case) was delivered to AFGL.

Year 2:

(a) The ASIP II was shipped from AFGL to Keo (towards the end of Year 1) for complete calibration procedures. An exhaustive calibration of ASIP was completed in Year 2, including absolute calibration, spectral sensitivity, vignetting, flat field, resolution, shutter characteristics, temperature effects, phosphor decay characteristics, and linearity with respect to gain and exposure. These results were documented in the report entitled "ASIP II - All Sky Imaging Photometer - Calibration Results" supplied to AFGL, and abstracted in Scientific Report #2 as Appendix III (AFGL-TR-88-0188).

(b) The new telecentric optics design (replacing the original bi-convex element with two plano-convex elements) was evaluated during the ASIP II calibration procedures. Resolution was considerably improved, especially near the edges of the all-sky field.

(c) The reimaging optics was redesigned, replacing the previous plano-convex + meniscus lens pair with a pair of achromats. This led to further resolution and vignetting improvements, and substantially corrected chromatic aberrations which had previously led to slight focal changes with different filters.

(d) A new filter wheel with capacity for five 3" filters was designed and constructed and delivered to AFGL. The new filter wheel controller utilizes a ROM that selects the shortest route (clockwise or anticlockwise) for changing filters in an arbitrary order.

(e) The lens coupling arrangement to the CCD chamber was modified to allow dry nitrogen flushing. This was necessitated because of window frosting problems in the (cooled) CCD detector housing. This problem first occurred in the humid environment encountered on a field trip to Puerto Rico. It was found on investigation that the CCD chamber was no longer holding a vacuum. Discussions with Photometrics Ltd.. suggested that this might not be unusual, and that we could not reasonably expect the chamber vacuum to hold for long periods. Because of the impracticality of taking vacuum equipment to the field, it was decided to implement the dry nitrogen flushing. Tests showed that the CCD minimum temperature was not adversely affected, and when frosting next occurred (again on a field trip to a humid environment - Wake Island), flushing clearing the problem in 20-30 seconds.

(f) A new image intensifier mount was designed and built to allow more convenient installation and removal of the image intensifier. This was found to be desirable during the calibration procedures, and we considered it would be desirable in general.

(g) A new optical design was developed (in cooperation with Dr. S. B. Mende of Lockheed Research Laboratories and the Lockheed Optics Design Group) for a high-resolution all-sky optics assembly. The design is specific to the all-sky primary lens, and so does not allow interchange of primary lenses. However it succeeds in giving very high resolution and freedom from chromatic aberration. Although not implemented to date, and requiring specially ground components, the design is available should a future need arise.

(g) During ASIP II calibration procedures, advantage was taken of the calibration set-up to carry out an absolute calibration of the AFGL secondary standard light sources. These calibrations were provided to AFGL.

Year 3:

(a) The optics improvements to the ASIP II (described above) were also implemented on the monochromatic intensified film cameras deployed at field locations at Svaalbad and Greenland. The optics/camera heads were returned to AFGL, and new Mamiya lenses, telecentric elements and achromatic reimaging optics fitted. The Automax cameras were also cleaned (especially removal of brush carbon in the motor) and lubricated as required.

(b) Prior to field deployment of the photometer system at the end of Year 3 (to Greenland), the telescope optics was aligned, and all filters checked for stability of passband since purchased. All were ok.

5.Hardware Development:

Year 1:

(a) Synch Stripper: A synch stripper was purchased and installed in the ASIP II system to correct synch problems in using the Photometrics scan converter boards to drive the RGB monitor.

(b) Image Intensifier Gain: Circuitry was designed and installed to allow the image intensifier gain in the ASIP II system to be controlled by the computer (in

addition to manual control). Gain steps equivalent to a photographic F stop (1,2,4,8) were preselected and calibrated.

(c) Image Intensifier AGC: An image intensifier power supply was modified to allow the monitoring of an internal voltage controlling the AGC circuit. Once the tube enters the AGC regime, it was found that knowledge of this voltage allows extension of quantitative intensity measurements by two orders of magnitude. For the present applications of this instrument to aurora and airglow, this feature would only be useful for the very brightest aurora.

Year 2:

(a) Controller: A new filter-wheel Controller was designed and constructed for the new 5-position filter wheel. The following features were incorporated into the Controller chassis:

(i) Filter temperature set and readout.

(ii) CCD temperature readout.

(iii) Image intensifier gain control (manual and computer).

(iv) Image intensifier AGC indication and monitoring capability.

(v) Image intensifier HV circuit continuity indication.

(vi) Manual/remote shutter switch.

(vii) Two auxiliary switches.

Incorporation of all the above functions in the one chassis simplified wiring and resulted in a more compact and professional instrument appearance.

(b) Time Base Corrector (TBC): Various problems arose with recording images from ASIP II (from Photometrics scan converter boards) onto the Panasonic optical disc recorder, in accepting images from the disc recorder into the Northwest Research Image Analysis System, and in using ASIP I images in the Northwest Research Image Analysis System. Analysis indicated that a time-base corrector (TBC) should correct these formatting problems, so a demonstration of the For-A TBC was arranged at AGFL, resulting in purchase and incorporation into the system. Besides correcting the problems mentioned, the unit has proved valuable in other video manipulation situations.

(c) Aircraft Power: A requirement arose for an addition frequency converter (400 Hz to 60 Hz) for the AIO, and a used unit was located and purchased.

(d) Foreign Power: To allow operation of the ASIP II in foreign countries, a power converter (220V/50Hz to 115V/60Hz) was purchased.

Year 3:

(a) A special 35mm camera system for the Northwest Image Analysis System was procured and installed, together with a video printer, to further facilitate image analysis being carried out with this system.

(b) Two new dual monitors were procured and installed to replace old units on the ASIP 1 system, as these old monitors were failing intermittently and displaying somewhat distorted pictures.

(c) A comprehensive study of optical disc technology was undertaken by Keo, for the purpose of updating recording facilities on ASIP II. It was deemed very desirable to replace the troublesome Kennedy recorders (two repairs needed last year) with the much smaller and lighter optical discs. As a result of this procurement research, the best type of optical disc, together with needed driver software, was identified. Quotations were solicited, and procurement of optical disc drives and software can now proceed as soon as funds are available.

(d) Keo worked closely with an AFGL engineer (Mr. Kuenzler) on hardware requirements for the new scintillation recording system. Circuit boards were laid out and fabricated, and required components procured, in time for assembly by a summer student working at AFGL. The assembly and testing of eight of these units for the field was nearing completion at the end of Year 3.

(e) Several hardware modifications were designed for the Photometer/Spectrometer system:

. A generalized 16 channel 12 bit Analog/Digital Converter Interface was implemented. This has been used for quick readouts of the mount Azimuth and Elevation positions, but can easily be expanded to read 14 other generalized axes.

. A generalized parallel port was provided on the back of the mount controller to interface to any parallel device in the field. This was designed specifically for the SRI Radar interface in Sondestrom, Greenland, but like the ADC interface, can be generalized to any 2-byte interface.

These two hardware modifications give the instrument a great deal of flexibility in being used in a wide variety of field situations.

. Trimming resistors were added to the filter wheel nulling circuits to decrease the current to the reflector switches, which solved a problem of unreliable filter nulling.

6.Documentation:

All ASIP II and Photometer/Spectrometer documentation has been periodically reviewed and updated throughout this contract, and three complete documentation sets maintained. Two have been provided to AFGL (one for field use, and one for lab use) and one set is retained at Keo.

7. Data Analysis:

Year 1:

(a). The P.I. participated with AFGL and the University of Texas at Dallas in a co-ordinated program to investigate conjugacy of equatorial airglow depletions (July, 1986; P.I. funding from National Science Foundation). Data from this campaign was used for the test and evaluation of the Northwest Research Image Analysis System. To this end, the P.I. traveled with Dr. E. Weber from the Ionospheric Dynamics Branch to Seattle to consult and work with Northwest Research. The equatorial data (from Hawaii and Cook Is.) was converted to geographic plots with data from both hemispheres plotted on the same map. Star fields were used to define camera look directions. Study of the one (weak) event selected showed conjugacy (or near conjugacy), and these data were supplied to Dr. B. Tinsley of University of Texas at Dallas. It is expected that the experiment will be repeated in the summer of 1990 or 1991 when the sunspot cycle peaks and more intense and frequent events are expected.

(b). Keo part-time personnel (Mr. Stephen Weisfeldt) worked on various data analysis tasks within the Ionospheric Effects Branch during the 1987, 1988 and 1989 summer months.

Year 2:

Keo personnel (Mr. Steven Weisfeldt and Mr. Michael Keane) continued to work with AFGL scientists as requested to assist in the operation of the Northwest Research Image Analysis System on a number of different projects. [Results from such analyses were presented by Dr. E. Weber at the CEDAR Workshop (Boulder, CO.) and the MIT Workshop in June, 1988.]

Year 3:

Keo Personnel (Dr. T.G. Tsuei and Ms. Colerico) worked with AFGL scientists as requested in assisting in the operation of the Northwest Research Image Analysis System on a number of different projects.

8.Mission Support:

Year 1:

Keo personnel participated in the following AFGL field trips during Year 1:

(a) Sonde Stromfjiord and Thule, Greenland and Norway (Nov., 1986):

Mr. James Moore (trip planning and mission director).

Mr. Cyril Lance (software and hardware engineer).

(b) Sonde Stromfjiord,Greenland (Feb.-March, 1987):

Mr. Cyril Lance (software and hardware engineer).

Year 2:

Keo personnel participated in the following AFGL field trips during Year 2:

(a) Thule, Greenland (Feb., 1988):

Mr. Michael Keane (software and operations support).

(b) Arecibo, Puerto Rico (March and April, 1988):

Mr. Michael Keane (software and operations support).

Other travel included a trip by the Principal Investigator (Dr. R. H Eather) to the CEDAR conference in Boulder, CO. in June, 1988 to attend imaging sessions, and also to attend a special meeting/demonstration at Ball Aerospace in Boulder on CCD imaging techniques.

Year 3:

Keo personnel participated in the following AFGL field trips during Year 3:

(a) Sonde Stromfjiord, Greenland (Dec., 1988):

Mr. Michael Keane and Mr. Cyril Lance (software and operations support).

(b) Duck, North Carolina (May, 1989)

Dr. T.G. Tsuei (software support).

(c) Arecibo, Puerto Rico (June, 1989).

Mr. Cyril Lance (software and operations support).

Other travel included a trip by the Principal Investigator (Dr. R. H. Eather) to the ISIE Conference at San Diego, CA in Dec., 1988 to attend a technical session on Image Intensifiers.

9. General Repairs, Maintenance, Procurement:

Keo responded to repair and maintenance requirements as they arose during the contract, and appropriate parts or equipment were purchased as necessary. A partial list, other than work specifically described in the above sections, includes the following:

- . CCD camera returned to Photometrics for in-warranty repair.
- . Procurement of DEC VT201 terminal.
- . Procurement of narrow band interference filters (9).
- . HV power supply repair (Westinghouse) for ASIP I.
- . Procurement of Sony video printer.
- . New image intensifier mount for ASIP II.
- . Test jigs for calibration procedures for ASIP II.
- . Special calibration/test jigs for optical testing and evaluation.
- . Procurement of 3 shipping cases for the ASIP II
- . Upgrade of optics for one of the film based imagers.
- . Procurement of hardware and software for the new scintillation data system.

. Procurement of a Varo image intensifier tube with special (blue sensitive) cathode and special (fast white) output phosphor.
. New drive motor for an Automax camera.
. New BCD digital clock to replace faulty field unit.
. Repairs of 3 photometer/spectrometer amplifier/discriminators.
. Numerous other minor repair problems.

10. Personnel:

The following lists Keo personnel who worked on this Contract :

Year 1:

Dr. Robert H. Eather - Principal Investigator
Mr. Cyril Lance - Software and Hardware Engineer
Mr. James Moore - Mission Planning and Director
Mr. Stephen Weisfeldt - Technician and Data Analyst.

Year 2:

Dr. Robert H. Eather - Principal Investigator
Mr. Cyril Lance - Software and Hardware Engineer (part time)
Mr. Michael Keane - Software Engineer (full time from Dec., 1987)
Mr. Steven Weisfeldt - Technician and Data Analyst (part time)

Year 3:

Dr. Robert H. Eather - Principal Investigator
Mr. Michael Keane - Software Engineer (full time through Dec., 1988)
Mr. Cyril Lance - Software and Hardware Engineer (part time)
Dr. T.G. Tsuei - Senior Software Engineer (April - Sept., 1989)
Ms. Marlene Colerico - Data Analyst (Sept., 1989)

Appendix I

Six Channel Photometer System

Operating System for Sondestrom Greenland 10/1989

KEO Six Channel Photometer System

User's Manual for Sondestrom, Greenland

This manual will describe the software and hardware system for the KEO Six Channel Photometer System as it was shipped to Sondestrom in October 1989. While the system seems operational at this point, there has been no exhaustive field testing and no doubt many modifications will be made in the field. This manual will describe how to use the present software, contain a description of the various software components, and give some pointers on using and modifying the software. The documentation may seem a little wordy for a user's manual, but keep in mind two things. One, it was written by me. And two, it's being read by you! By this I mean that I know that you can learn about the software by actually using it without a manual. The purpose of this manual, then, is to fill in the gaps completely, and to have you really understanding the system's capabilities and problems. Skip over programmer's notes if you are not a programmer. Skip over my dissertations on what could be, if it's driving you crazy and you just want to know how to turn the thing on.

This generation of the Photometer software comes from the original software that John Hogan started back in 1984. I have changed all the names of variables and routines to be somewhat pneumonic and have tried to reduce the level of inefficiency in the methods used. There are still a great deal of quirks to the system, and probably some inconsistencies. However, I hope the present system provides a working skeleton for you to use in Sondestrom. From being out in the field, you will learn exactly what you WANT from such an instrument. Hopefully, you will be able to make some of these changes. I would encourage you to keep a section in your notebook that deals with a wish-list so that when you return, we can plot a course for further development.

The evolution of this project has dictated the present design and as it is rather complicated, care should be taken when modifying even small things. Presently, the system utilizes the system CPU power exhaustively during the period of data acquisition, a period when there are interrupts (some in FORTRAN) and data recording occurring. There are many neat things that one would like to have done in real-time in the field, such as screen displays. These are very easy to implement in software, however, they could have tragic results (results without obvious manifestations -- the worst kind) for the accuracy of the data. Thus, my caution is that, while the computer is capable of a great deal more than it is presently doing, quick modifications should be discouraged until the operator really understands the anatomy of the operating system: One would not want to do cosmetic surgery before one understood the nervous system!

The thing that I tried to guarantee with the system before it was packed up, was that it would take data reliably at a constant rate, and that the various controlled axes moved with a constant rate. At any time, one can verify these criterion by reading back data of predictable results, and to check the motion of the axes by measuring the cycle times of the controller pulses. I suggest that these things be done before and after each software modification.

The Basics -- Using the System

The use of the Photometer system is pretty self-explanatory, but I will go through a typical operational set-up to fill in some of the gaps. Unfortunately, at the time of this writing, the system was being shipped to Greenland, so some of the details may be a little distorted by my memory!

The most basic thing about using the system is turning it on! While this seems kind of silly, there are a couple of points to make. The first is the issue of power. Always make sure that the

Lambda Power supplies are isolated from the rest of the system using the RFI filters, because the system is very sensitive to the switching power supply noise. You might also be affected by noise from the Radar operation, and this must be checked. The hard disk should be turned on first and allowed to run up to speed and the terminal should be turned on and have finished its self-check before turning on the computer. If in this state, the computer will automatically boot from the hard disk. If not, the computer will sit in a HALT state and the run light will not be on. To boot the computer manually, flick the BOOT switch up momentarily. After a few seconds, the RUN light should turn on on the computer and RT-11 System messages should start appearing on the screen.

You can also reboot from the terminal, if it was remote from the system, by either hitting the **BREAK** key (at which point you halt operation of the system and an @ appears on the terminal), or if the computer is already HALTed (there is already an @ on the terminal) and giving the boot address of the system and the GO command:

@773000G

After a couple of seconds, the RT-11 System messages should start appearing on the terminal. At any point during operation of the software, you can HALT and RUN the system by using the **BREAK** key. An @ should appear and to continue with the software, you just need to type:

@P

This is very useful for debugging and if you are planning to program the system or debug the hardware, you should try to become familiar with this operating mode by reading in the RT-11 manuals.

Hopefully, the computer booted with no problems and now you are in the RT-11 operating system. The computer should boot into the Foreground/Background system called RT11FB.SYS on your disk. It will prompt you for the date and time. This is very important to enter accurately and some care should be taken as this is the time base for your data. A typing mistake here could cause incredible headaches back home at Hanscom. I am not sure how accurate this clock is in real time, and the drift of the clock should be monitored very carefully.

The hard disk is divided into three equal partitions. Under RT-11, they are called DL0:, DL1:, and DL2:. The system files for RT11FB and RT11SJ (Single Job) and the original Photometer/Spectrometer files are stored on DL0: which is where the computer boots to. DL1: contains all the system files for using RT11XM, which is an Extended Memory operating system. This system will become very useful if you want to eventually expand the program size beyond the 32Kword limitation. However, right now this system is never used.

DL2: is reserved for this photometer system. There should be NO other files put on this disk. I can't stress this enough. A sloppy use of your hard disks is the easiest way to cause errors, confusion, and unseen bugs. DL0: is pretty sloppy leftover from the previous programmer. Names are inconsistent and there is a lot confusion when doing a directory listing. I would like to avoid that with DL2:. When you boot the computer, DL2: is automatically defined as the default disk (DK:). Thus, you are already residing on the correct disk partition to run the Photometer software.

The first thing you need to think of is where you want to store your data. Presently, it seems possible to write to either Magnetic Tape, or a file directly on DL2:. The first seems more reliable and permanent, however, there will be times when you do want to use the disk to store data. One example of this might be that you are in the middle of taking data on a magnetic tape and you want to do something different just for a second, and then return to your normal operating mode. Instead of

having a weird file in the middle of your data tape, and instead of rewinding the tape, mounting a new one, then rewinding that, and mounting the old one and returning to the middle of the tape, it would be much easier simply to write this new data to disk and make a good note in your log. This file can then be put on tape at a later time.

To use the magnetic tape, there is a little prep work to do. If the tape is partially used, and you are just adding more files to it, you simply have to mount the tape, making sure there is a write ring in it, put it ON-LINE, and start the Photometer system software. If the tape is new, you need to initialize it under RT-11. This creates a system directory file on the tape to keep track of files. Once this is done, the tape looks just like other devices to the operating system. However, there are many nuances to using Magnetic Tape under RT-11 that should be read about in the manuals (particularly under the system utility PIP). To initialize a tape, you can use the RT-11 command:

.INIT MT0: (The . is the system prompt)
INIT/Are you sure? **Y**

When the computer prompts **Are you sure?**, don't rush over this step. If you by accident typed DL0: instead of MT0:, you would do some serious damage. So take the second -- did I type MT0:? Uh-huh, OK, I'm sure...

The tape would then be initialized as an RT-11 system tape. I have included a command file on the system called **INIMT.COM**. This command allows you to give the tape an owner and a tape name. Since there will be lots of data being taken out in the field for different experiments, it will probably be useful to utilize this automatic labelling to keep the data a little bit more organized. To use, simply type:

.@INITMT

The system will then prompt you for the tape labels, and then initialize the tape and create a new directory on the tape. I suggest always using this function and finding a consistent method of labelling the tapes, it will save you much time back at Hanscom.

If you want to use the disk for data stoarge, there are a couple of prep things to do to make sure everything runs smoothly. The first and most important is to keep the disk immaculately clean and organized. Miscellaneous test files that aren't documented, data files for test runs, files with funny names have no place on DL2:, they should be deleted or moved onto a floppy disk. The next thing to do is to clean up where the remaining files are on the actual disk physically. This allows the greatest number of contiguous blocks for the data file to be written and thus reduces problems related to this. To achieve this, type:

.SQUEEZE DL2:
Squeeze/Are you sure? **Y**

Again, this is a crucial operation. Do not rush it. The system is asking you if you are sure for a good reason. If you were to accidentally type **SQUEEZE DL0:**, the operating system might move the system files around and accidentally move the boot file from the proper sector on the disk, making it unbootable. This would be a very regrettable error, and if it happens, make up another reason for the system being down (a Huuuuge power surge, must of been something Cyril did, etc...).

Once the system is done with this process, you are ready to record data to disk. It should be noted that for small test files, this is not really necessary and that since the process takes a little bit of

time, you will probably not always want to do this. However, if you are going to seriously take a decent amount of data onto the hard disk, I would recommend this step strongly.

So, at this point, either the Magnetic Tape, or the Hard Disk is prepared to take data. So lets enter the Photometer Operating System:

.@PH

This starts running the **Photometer** software. The system first asks you for the data storage file:

Enter the name of the data file: **MT0:89SN14.DAT** (or **DL2:89SN14.DAT**)

This is an example of a file name: 1989 - in Sonde -- tape #14. I'm sure you will come up with your own method for labelling data files. Once this is done, the screen will clear and the header will appear. At this point, you should make sure that the date is correct, and that the name of the data file is correct. This is a good place to point out a problem and limitation with the software. Presently, there is no real file handler and this severely limits what you can do. To change the data file, you have to leave the system. If you make a typing mistake, and need to create a new data file, you have to leave the system and start again. It works, but it is clumsy and can lead to some problems.

It should be noted at this point that at anytime during operation of the photometer software, you can abort the program by typing two consecutive ^C's. Undoubtably you will have to do this as problems arise, or you entered a parameter wrong. Unfortunately, this could have drastic repercussions for your data file, as the file will not be closed. On disk, you will loose the files, but on tape, the directory will not be updated, and thus the tape could have some problems for the next file. If at all possible, instead of aborting the program, you should try exiting it from within the software, even if this will take an extra minute. Your next question will be, what if it aborts on its own? (And it will, say by typing an integer in, when it wanted a floating point number.) This will cause you unending frustrations and it's a real problem. You will undoubtably dub this system many new names as this problem keeps re-occurring. The only suggestion right now is to be very careful with what you type, make sure you know what you are doing, avoid typing ahead, and always, always take good notes of what you are doing. Hopefully, a better scheme will be developed.

Programmer's Note#1: How can one handle data files from within the system? How can one handle system aborts better?

A new programming section could be added that would be similar to the Mount or Filter Initialization routines, that would handle all file operations. Things such as opening new and old data files could be done from within the photometer operating system. This would greatly increase the efficiency of the operating system and avoid a lot of frustration with dealing with data files. In addition, this software could include provisions for writing the parameter files to tape as well, so that there is some permanent record of what the operational mode of the system was when data was being recorded -- What was the velocity of the Azimuth axis? What was the filter integration time? My notes are a little messy. This software could be integrated into the same memory overlay segment as the other initialization routines and thus add no new memory overhead to the system.

A way to avoid system aborts due errors associated with data entry and typing errors, is to develop a photometer system error page that the software can abort to when it encounters these problems (i.e. see the ERR option in the FORTRAN READ statement.

Next, the Photometer software starts the initialization routines. First to be set up is the Filter system:

Would you like to calibrate the filters?**Y**

If you want to calibrate the filters type a **Y**, otherwise just hit a **<RET>**. A filter calibration takes a while and is used to find the peak transmission angles of the filters automatically. First, the software nulls all the filters. If this is not accomplished, a system message will result that not all the filters are nulled, but there is no further provision and the software keeps operating. This is obviously an operational problem and can be changed with some simple software modifications, however, if this happens, you will need to abort the program, figure out why the filters are not nulling (maybe try it again) and then restart the whole process with an **@INIMT** (if necessary), and then a **@PH**. (Now, you begin to see the reason for the above note!)

Programmers Note #2: What if filters don't all null?

A simple modification to FNULL.FOR could pause the software if not all filters are nulled, and ask the operator if they would like to try again. Are the filter motors turned on?, Are they plugged in? Oh yeah, I forgot to plug them in, OK let's try again -- ah they all nulled, good, lets continue, etc...

Once the filters are nulled, the software immediately begins the calibration routine. Filters are stepped and then the counters are integrated for 20 msec and recorded in a data array. The whole filter cycle is repeated ten times to provide a good average. These values are then used to find a maximum value for each filter and it's position for that value is recorded and noted to the operator. Finally, this data can be recorded onto the data tape or to a file on disk with the prompt:

Please enter a calibration file name: **DL2:HIST24.DAT**

In case the operator is unsure of whether these values are accurate, you are given the option of doing another calibration to make sure the positions are repeatable. Keep in mind that there are 100 steps per filter cycle and that the filter wheel moves about 10 degrees. The relationship between steps and degrees is not linear because the filter wheel tilts with a cam device. There are many more degrees/step at the limit of the tilt (10 degrees) than there are near the 0 null position.

Programmers Note #3: What if I need a different integration time?

There are couple of modifications that could be made to FILCAL.FOR, but the most important would be to give the operator an option for integration time. This could easily be modified by using the same method as in the program TSTCNT.FOR as they have the same format. In addition, there is no real need to take the average of the histogram (dividing each value by 10) and this part of the program could be commented out. The calibration routine previously would automatically re-null the filters and position them to the maximum, but then I decided the operator might want to take some tilting data first so I commented this out too.

Once these filter positions are determined, they should be written down in your data log and checked from time to time. Are they consistent? What is the filter temperature for each data point?

The next step is to set up the operational mode for the filters. The software gives you the options of having each filter either tilt or remain at a fixed position (probably the one calculated by the above calibration). Right now, the handling of these parameters is kind of sloppy and there are no elegant menus such as in the KEO system, or a McInctosh, but these could be added to make this easier. There are presently four filter rates allowed to be used: .120, .5, 1., and 2. seconds per revolution. Thus a filter rate of .5, gives us 50 steps per second, or a 20 msec integration time. It will probably be necessary to add more options and you want to consider having a much more flexible method of determining filter rates.

The current stepper rate (rps) is .500
Enter new rate (.125,.5,1.,2.) or hit return: **1.**

Now that the filter rate has been selected, all filters are assumed to be tilting and the operator can set up the fixed filters and their fixed positions:

```
Current Motor Status: 1-T  2-T  3-T  4-T  5-T  6-T

Enter the number and step position for each of the fixed motors: 2,67
Filter #2 set to 67. Enter next selection: 4,24
Filter #4 set to 24. Enter next selection: <RET>

The following motors are still in tilting mode
  1  3  5  6
```

The operator in the above example, set filter 2 to position 67, and filter 4 to position 24, and left all the other filters in tilting mode. Next, the computer asks for the integration time of the photometers. The system chooses integration times based on filter steps. If the filters are stepping at 50 times a second, and you integrate for 4 filter steps, then your integration time is 80 msecs. Even if none of the filters are actually tilting, the computer still operates on this time base. Thus, in the above example, every 20 msecs, the computer interrupts and sees if the filters need to be stepped, or the counters need to be read. Thus, to make the CPU as efficient as possible, if none of the filters are tilting, you would want to integrate for every filter step, and have this imaginary "step rate" (even though no filters are stepping) be equal to your desired integration time. This limits the amount of times the computer has to interrupt, and thus decrease the CPU overhead.

```
The integration period can range from 1 to motor steps per period
The current number of steps per period is 4
Enter new selection: 1
```

Let's say in the above example that you had been tilting the filters once every 2 seconds (filter rate = .5) and that you were integrating every 4 steps giving you an integration time of 80 msecs. If you now wanted to set all filters at the FIXED position determined by FILCAL, and integrate for 80 msec, you would ideally want to set the integration period to 1 step per period and have the filter rate be .125 revolutions per second. This is allowed in the above system, but I think you can see that there could easily be some problems related to this way of setting up the filters (see programmer's note #4).

The photometer system then writes all this parameter information to a file called DL2:FILTER.DAT. Keep in mind that every time you do this, the old file gets written over so it is CRUCIAL that you keep a log of these parameters at all times. A better way to handle this is mentioned in programmer's note #1. Another interesting point is that these files are not presently read upon entering the filter initialization, so that you need to re-set everything everytime to run the program. This, also, is unnecessary and is addressed in programmer's note #4.

Finally, the computer nulls all the filters. Again, if not all filters were nulled, the computer continues on. This is a potential problem and can easily be avoided (see programmer's note #2). Next, the computer positions the fixed filters to their requested positions:

```
*** All Filters have been nulled ***
Filters nulled: 1 2 3 4 5 6
```

If not all filters were nulled in 200 steps, the nulling routine notifies the operator and continues on:

```
***** NOT ALL FILTERS WERE NULLED *****
Filters nulled: 1 3 4
```

Next the filters are all positioned to their proper positions for fixed mode operation. In the above example:

Filter #2 positioned to 67 motor steps
Filter #4 positioned to 24 motor steps

Programmer's Note #4: How can I improve the filter initialization process?

One obvious improvement would be to change the routine and all other initialization routines to a method similar to that of the ASIPII system, where all parameters are displayed in a logical format, and you are given the option to individually (or as a group) modify the parameters you want. Once modified, the parameter is updated in the screen display of this set. In addition, this could allow you the option to have different parameter sets stored in different files on disk and the operator could have the option of opening up different parameter sets (say from last night's run). This is another universal scheme that could be implemented. Presently the software writes a parameter file into FILTER.DAT, but there is no way to store specific parameter files except by using the ***RENAME*** *command under RT-11. Also, this set could be read in at the beginning of the filter initialization (FILINI.FOR) as most likely, the modes will be very similar every night of data acquisition. Also, it would be nice to have a more flexible set of filter rates, and integration times. This would be very easy to change.*

Programmer's Note #5: ASCII Status words .vs. Binary status words

A universal problematic approach in this software is that John Hogan loved to use ASCII byte flags to store the state of the system (i.e. if a filter is tilting, this is stored as a 'T' in the BYTE FCYCLE(i) for filter #i). This is a waste of memory, and is also very inefficient to program. A better way to implement all these system states would be through binary status words, where each bit represented the state of an axes (i.e. BYTE FCYCLE: bit 0: 1=tilting, 0=fixed, bit 1: etc...) These status words could be stored in an absolute memory location using a MACRO .ASECT and thus at any point, the computer could be halted and the status of the system could be PEEKed as simple binary flags.

The filter initialization is now complete and the operator is ready to set up the mount parameters. These control the motion of the Azimuth/Elevation mount that points the telescope to different parts in the sky. There are 50 steps per degree resolution in the stepper motors that control each axis and the computer keeps track of the azimuth and elevation by counting the number of steps it has moved since the mount was moved to the zero location. To double check that the mount is operating carefully, there is an analogue readout of the mount position. When at rest, this is accurate to within +/- .5 degrees, but when the mount is moving, there seems to be some sort of lagging between the computer's mount position and the analogue position. This needs to be looked into in more detail, but it never seems to more than 2-3 degrees, and thus is not outside the telescope's 5 degree field of view.

Again, this software has many of the same characteristics of the filter initialization routine.thus you will run into a lot of the same problems. The first thing the mount initialization routine does is to read in the parameter file MOUNT.DAT. This has the last available mount parameter set in it. If it can't find such a file, it will create one. Once the initialization is completed, the new parameters are updated into this file. Again, this creates a similar problem as with the filter parameter set (see programmer's note #5).

Allowed mount MODES are POINTING or SCANNING
The current mode is POINTING. Change to **SCANNING**

This sets the mount up to either point at a particular spot in the sky or to scan through two end points continuously back and forth. A third mode, slaving to the radar, should be added (addressed in an appendix). Unfortunately, you have to type in the mode instead of just selecting it using an easier method (see programmer's note #5). If the mode is scanning, the end points are requested and the scanning rate. If the mode is pointing, just the pointing location is requested. In the above example:

The current ELEVATION scanning rate is 2.0 degrees per second.
The new scanning rate must be 6.0 or less degrees per second.
Please enter new scanning rate: **<RET>**

The current AZIMUTH scanning rate is 3.0 degrees per second.
The new scanning rate must be 6.0 or less degrees per second.
The new rate is: **2.0**

The current start ZENITH angle is -20.0 degrees
Please enter new ZENITH angle or hit return: **-45.0**

The current stop ZENTIH angle is 36.0 degrees
Please enter new ZENITH angle or hit return: **45.0**

The current start AZIMUTH is -130.0 degrees
Please enter new AZIMUTH angle or hit return: **-80.0**

The current stop AZIMUTH angle is 129.0 degrees
Please enter new AZIMUTH angle or hit return: **80.0**

The computer then writes these parameters to a parameter file MOUNT.DAT. In the above example, I set the mount to scan at 2.0 per second (100 steps/second) between points (-45.,-80.) and (45.,80.). Notice that during the scan, both the AZIMUTH and ZENITH should cross their zeros at the same point. In other words, the scan goes through the point (0,0). The other thing an operator would notice is that there's a slight problem with this. How can I be moving the mount smoothly from point A to B with the same scanning rate (in the above example) if the distance of the two axes is different? Obviously, you can't! The software looks for the longest axes and divides down from there, so, in the above example, the AZIMUTH scanning rate is 2.0 degrees/second and the ZENITH scanning rate is divided by 160/90 (So in this case the real ZENTIH rate is 2*(90/160) = 1.125 degrees/second). This could be corrected in the new system -- how about asking for an angular velocity in degrees/second and then having the software figure out what the major and minor axes velocities are?

Once these parameters are decided the mount needs to be checked and set to the starting position. When first running your system, you will probably want to know if the mount is working correctly, as you will probably not be able to look at it visually. The software checks the end limits of the mount and then moves to the physical zero position of the mount and sets the software position variables to zero. Once this has happened, you know that the mount is working, the computer knows where it is and you can move to the start location and start taking data.

Since you will undoubtedly will be changing parameters here and there, eventually you will get tired of waiting for the mount to check for all it's limits and then zero and then move to start location. I certainly did. So there is a provision to just zero the mount very quickly and then move to the start location. This assumes that computer accurately knows where the mount is. If it doesn't there is a chance this will not work and you could run into trouble. I have found that it works pretty well, but since it was written an hour before I had to have the software finished, it obviously has not been fully

tested. My suggestion is that if you are not sure, and have time, check the limits. If this quick zeroing scheme seems to work well, use it when you don't have a huge data file open (and thus don't have that much to loose). If it seems to always work, take your chances!

Would you like to do a quick zeroing of mount? **N**

The computer will now step the mount to the CW limits and then to the CCW limits and then find the zero. After these steps, the mount will slew to the starting location and the computer will wait until the operator wants to start the data acquisition cycle.

*** Checking Mount Operation ***

Checking CCW limit indicators
Azimuth CCW limit is reached
Elevation CCW limit is reached
MOUNT clock is working

Checking CW limit indicators --
Azimuth CW limit reached
Elevation CW limit reached

Moving to locate ZERO
Azimuth ZERO found
Elevation ZERO found

Moving to START elevation and azimuth

PAUSE Hit <RET> to start scanning **<RET>**

If at any point, a limit is not reached, or there is a problem (mount is off-line) the software will pause and let you address the problem. There is a time-out limit in searching for the limits and the zero in case the mount gets stuck in a position (i.e. the fiber optics cables have snuck up another 2/1000 of an inch, or there's a physical obstacle, a stepper motor board burned out, etc.).

If you used the quick zero option, the mount would have just slewed to the zero position and once found, moved to the starting position. This software reads the present analogue position of the mount (because you can't assume the computer position is valid) and based on the position, issues a set of movements to get the mount to the physical zero position. It is a very simple approached and could probably be modified to be more time efficient.

Programmer's Note #6: How can I improve this initialization?

As with the filter initialization, there are the obvious problems with how the parameters are displayed and accessed. Right now, everything is linear and to change something back two steps, you would have to go through the whole process again which is very time consuming and very frustrating for a simple typo. So, a parameter display and simple editing feature would be nice and could be coupled together with the filter routines for compactness. Again, there's the problem with redundant flag variables and flags in 4 Byte ASCII words. Again, this is a waste of space and a waste of CPU time. These could be changed over to binary status words located in absolute memory along with the filter flags. In addition, the option of different parameter sets could easily be implemented.

Once, you have finished with this section, the system just waits until you would like to start acquiring data. This option is so that you can synchronize the data run with a real-time clock such as

the Kinemetrics satellite clock. It would probably be a good idea to note exactly what time you start and end each scan with this clock in your log book as I have no idea how accurate the CPU keeps time. This will need to be looked at very carefully.

You are now taking data!!!

Programmer's Note #7: What's the computer doing right now?

At this point, it would be nice to know exactly what the computer is doing by looking at the terminal. Displays of mount position, filter wheel activity, present count rates of the photometer tubes, what data buffer is being read, etc. By having this on the screen, an operator could feel comfortable operating the system with a remote terminal, say down in the radar control room. At this point, there is not enough CPU time to write to the screen with terminal interrupts because so much time is spent processing these FORTRAN interrupts to move the mount and tilt the filters, and every 1/2 second or so write a data buffer to the tape.

These are not insurmountable problems, interrupts can be simplified and written in MACRO, data buffers can be adjusted to make this more efficient , a tricky low priority terminal display interrupt can be written to update the screen only as time permits. However, for right now, the operator is stuck with having faith that the system is working as envisioned. This means that the operator will need to take doubly good notes, and always be checking their results.

A very easy modification to the existing software would be to add another option when you finished a data taking run and are still in the software. You could write a piece of code that just displays all the pertinent parameters such as the mount positions, filter positions, the last 10 data points, the time and data of the last data buffer, etc. Once satisfied, you could return and the computer would continue to take data as if nothing had happened. You could even modify this so that every minute during the data run, you would stop the run (STPINT), display the latest data buffer and continue on (GOINT) and this would happen automatically and very quickly (less than a second).

The mount should now be scanning at the proper rate to the limits you requested, or it should be pointing at the right place if possible. Check these things. Is the mount moving fairly smoothly? Check that the filters are tilting by putting your fingers on the filter shafts. Are filters 1,3,5,6 tilting and filters 2 and 4 stationary? If you have confidence in the system, then you could leave and it would run indefinitely until the data storage device got saturated. Unfortunately, this would crash the computer and you would loose your data, so it is important to be aware of how long you have. The day before the system was packed, the system ran for 8 hours taking 10 data points/sec without a problem and the magnetic tape was almost filled up.

At some point during the data run, you will want to stop the process and either close the file and create a new one, call it quits for the night, modify a parameter, etc. To do this, you must type consecutively:

CO<RET> (short for **CO**mmand)

The reason it is so complicated to stop the interrupt is that this instrument was initially supposed to be installed on the aircraft where space is limited and it would be very easy to brush against a keyboard that was exposed to the walkway. Thus, miscellaneous characters are ignored. Once this has been typed, the computer will prompt:

Command requested:

Presently there are only two commands: **QUIT** or **INIT**. QUIT stops the data acquisition process, closes the data file and exits from the software. Perhaps a better option would be to close the file and then ask the operator if they want to leave the photometer system or to open a new data file, change

parameters, etc. INIT goes through the whole system initialization routine again keeping the data file open. This is a little dangerous, especially if you don't take good log notes, as you can change modes drastically inside a single data file. Hopefully, each buffer has enough information to analyze the data correctly, however, it makes me nervous. This scheme should be re-thought and changed as per programmer's note #1.

You have now used all the basic photometer system commands. Use in the field will show you how to really utilize this system in its present configuration. The present system is clumsy, frustrating, and inefficient, however it works (last time I checked!). Hopefully, it will be sufficient for this campaign.

Photometer Utilities -- Checking things out

Since the run time checks for the photometer system are so limited, I have written a few simple routines to help analyze the performance of the software and to help with debugging software and hardware. There are millions of these routines that can be envisioned and written. I encourage writing more that are really useful, and carefully documented. I discourage writing lots of little routines with funny names on DL2: that do this and that with no documentation.

Probably the most important thing to check right away is "Are the photometers counting photons and can the computer read this?" The first part of the question can be addressed best with an oscilloscope and the amplifier/discriminator documentation describes this well. The second half can be looked at using the software **TSTCNT**. To use this software, simply type:

.RUN TSTCNT (The <.> is the RT-11 System prompt)

Enter integration time (in 20msec units): **25**

The screen will then clear and the six different photometer channels will integrate (in the above example) for 500 msecs (1/2 second) and update the counter values to the screen. This is an extremely useful piece of software. First, it can be used simply to test that the photometers are working correctly. Are they all counting? Are the discriminators consistent from channel to channel if you change which photometers go to which channel? Is there a lot of noise coming into the counter boards with the HV off, with the 6V power supply voltage off? What happens if I turn up the discriminator some to reduce noise? Am I decreasing the count rate?

An other useful aspect of this routine is to see what kind of exposure you want to use for the particular night conditions. Unfortunately there is no provision to move the filters to a particular point in this routine, but it could easily be added. For a 1/2 exposure, am I getting enough counts, how about a 20 msecs exposure? Am I far enough above noise to get good statistics? All these questions and many more are answered much easier with the use of this utility.

Another very useful thing to do is to read a record from a data file. This uses the primitive utility **READTP**. To use this, make sure your data medium is loaded, and type:

.RUN READTP

Enter the name for the data file: **MT0:89SN14.DAT**

The software will then open this data file and read the first record and then wait to see if you want to read another record. If you do, you can hit any key except an N. A data record will be displayed as:

```
Date: 23-Oct-89                          Time: 10:23:45
AZ/EL:  Mount Pos:  -34.5  134.2        Radar Position: -31.0   138.4
Azim: -35.6  Elev: 132.9    Filter Positions: 23  67  58  24  67  47
123465    234345    456567     45345    456546    45345
435345    345345    435345     67675    344545    34545
324344    456556    367787     75655    245454    25467
234458    578987    245656     85656    132567    23546
229854    467876    220897     93473     93467    19347
192367    410920    197677    108787     83467    23090
148787    387873    238746    138783     76236    33897
 98787    834887    345464
                          ...etc.
```

There will be 12 rows of data because each record is the result of one data buffer being read onto the tape. You should check that the date and time is correct. If you are taking data 24 times a second, then the time should jump one second every 2 records, 4 seconds every 8 records. Is this happening? Is the year correct and the day? The second line displays the analogue value of the mount position at the time the data record was written. Is it within a reasonable range of the Azimuth and Elevation listed in the next line? Is the Radar position reasonable? If you suspect a problem with either of these, there is another utility **TSTRD** that can check this.

Next check that the computer mount positions are reasonable. If the mount Azimuth is moving at 2 degrees/second and the Zenith at 1.125 degrees/second, and each record represents 1/2 second of data, do the axis change 1 degree/record and .5 degree/record accordingly? Is there a software problem here? (I sure hope not, but you had better check!) Are the filter positions moving correctly? Is filter #2 at position 67, and filter #4 at position 24 for every record? Are the other filters tilting? If there is one filter step per integration, then every record should represent 12 filter steps. Are the filter positions at the next record 12 more than they are for this record? Does the filter position wrap around properly at 100? If there is a peak for filter #3 at position 46, and you have a continuum, does that peak stay at position 46 or does it jump 4 positions forward every cycle? (Hope to god, it don't!)

These are just a few of the questions to ask when looking at the data record to check that everything is working correctly. As you can see, it would be very useful to check this while the program is running (see programmer's note #7).

Another very helpful utility will be **TSTRD**. This utility reads the radar port and the analogue mount position 5000 times (an antiquated feature that I used to time this routine -- notice that it takes only 200usec) and then displays the converted values. This will be especially useful to get the RADAR interface hooked up. It might be useful to put a loop in there and take out this 5000 repetition loop to make this more useful. That way you could monitor the radar as it scans. Also, a loop would be useful if you need to recalibrate the ADC circuit that reads the mount position (this is described in the Mount Controller documentation envelope). There is a test plug built for the radar interface that puts a bit pattern into the DRV11-J interface that reads this port. This plug should give a radar position of:

Azimuth: 119.0 Elevation: 238.0

If this is not the case, there is something wrong with the interface.

It is probably worth going over quickly how each of the 3 positions are stored since they are all different(!):

	Azimuth	Elevation
Computer Mount Position:	AZ/20.0	EL/20.0
Analogue Mount Position:	(AZ*(360/4095))-180.2	(EL*(360/4095))-180.2
Radar Mount Position:	AZ*1.4	EL*1.4

At first glance, you would say, "geez, why don't you store them all with the same units, this is so confusing" and the answer is that there's not enough CPU time to do this arithmetic while the interrupts are going and data is being taken. Floating point arithmetic is time costly. This should give you an appreciation of just how tight the computer is for time.

The final utility written is **RDHIST** and is used to read files written by the FILCAL routine. This works the same as READTP, but scrolls through the Histogram values for each filter positions and then displays where the peak was for each filter. This information will be useful to compile when you get back at the lab. It would be useful to make a note of filter temperatures for each Histogram file in your log book as then when you come home and plot these out, you can make sure that the peaks remain constant with constant temperature and that the shape of the curve looks correct (i.e. two peaks) and a low point at maximum filter angle. To use the utility, type:

.RUN RDHIST

What is the name of the Histogram file? **DL2:HIST24.DAT**

PAUSE Opened file **<RET>**

Programming Hints for RT-11 Command Files

This section is not meant to replace the operator reading through the FORTRAN and RT-11 manuals to get a feeling for how to fully use this computer, but it will list the command files that I have on DL2: to help you understand how the software is put together and how to modify it.

The command file **PHTSYS.COM** shows the system components well and how to build a system. To run it, type:

.@PHTSYS

The command file will execute the following commands:

```
FORT/NOLINENUMBERS/CODE:THR FILINI, MNTINI,FILCAL,SONDSY

MACRO PRMETR

LINK/PROMPT/MAP:SONDSY SONDSY,PRMETR
PHTLIB,DL0:FORLIB
FILINI/O:1
MNTINI/O:1
FILCAL/O:1
```

The first line of the command file compiles all the major FORTRAN components of the Photometer system. The options NOLINENUMBERS and CODE:THR are used to reduce the amount

of physical space each routine takes once compiled. FILINI.FOR has the filter initialization routines. MNTINI.FOR, the mount initialization routines, FILCAL.FOR, the filter calibration routines, and SONDSY.FOR, the main photometer system routines.

The second line of this command file compiles the MACRO file PRMETR. This is an assembly language routine that contains all the global parameters.

Finally, you need to link all the elements of the program together into one large binary file (SONDSY.SAV) which will be an executable file. This LINK command show how this file is structured. The main components to be linked are SONDSY, PRMETR,PHTLIB,FORLIB, and the memory overlay segment which contains the three initialization routines. PHTLIB is a user created library of routines written in both MACRO and FORTRAN that will be discussed in more detail in the next section. FORLIB resides on DL0: and is the system supplied FORTRAN library which contains the executable code for all the FORTRAN routines. Finally, the memory overlay segment is a method used to save space. Since none of these initialization routines are used at the same time, or in real time, it makes more sense to create an area in the code where they can be written in and out of memory as needed. This is described in detail in the LINK documentation under RT-11.

If you just wanted to link the routines together (maybe you had changed something else), you can use the command file:

```
.@PHTLNK
LINK/PROMPT/MAP:SONDSY  SONDSY,PRMETER
PHTLIB,DL0:FORLIB
FILINI/O:1
MNTINI/O:1
FILCAL/O:1
```

Once, these steps have been taken you have a file called SONDSY.MAP which shows you exactly what the binary image looks like. This can be very helpful during debugging. In addition, you have an executable file SONDSY.SAV which can best be run by typing the command file:

```
.@PH

.SET USR NOSWAP
.RUN SONDSY

Enter name of data file:
```

Once you have typed this, the photometer system is running, and you are ready to continue as in the first section of this manual.

Another very important command file automatically builds your user library PHTLIB.OBJ. This is done by typing:

```
.@PHTLIB
```

This command file will go through all the library routines, compile them either from their FORTRAN or MACRO source code into binary files (.OBJ) and then create the library using the LIBR utility:

```
.LIBRARY/CREATE/PROMPT PHTLIB  INTRUP,ININTS,INICLK,CLKCHK
INTIMR,SETCLK,ARMCLK,MNSTAT,SPTMNT,SLEWIT,FILMOT,PHTCNT,
RESETM,MNTZRO,MNSLW,FNULL,FPOS
//
```

All this different files are compiled together into one large binary file called PHTLIB.OBJ which is then used when linking to the photometer system. You should read through the LIBR utility if you want to fully understand how this works. If you want to modify one particular element of the library such as FNULL.FOR which could use some modification, there is an easy way to do this without having to compile everything and re-create the whole library (this is the purpose of a library). I wrote a command file for the modification of FNULL to show you how this is done. Any other file in the library system can be changed similarly.

Say you have finished editing FNULL.FOR to change how it handles a nulling error, and you want to update the library. Simply type:

```
.@FNULL          (Could be called something slightly different!)
FORT/CODE:THR/NOLINENUMBERS FNULL
R LIBR
PHTLIB=PHTLIB,FNULL/R
^C
@PHTLNK
```

Now the new version of FNULL.FOR has been compiled and added into the library, and the whole system has been linked again and you are ready to run the photometer system by typing **@PH**.

Since you will be modifying SONDSY.FOR quite a bit, there is a command file to go through this process automatically. You could write a command file similar to this for PRMETR.MAC, FILINI.FOR, FILCAL.FOR, or MNTINI.FOR as well, but you don't want to have too many command files lying around your disk. However, this command file should show you how do such modifications:

```
.@FS
FORT/CODE:THR/NOLINENUMBERS SONDSY
@PHTLNK
LINK/PROMPT/MAP:SONDSY  SONDSY,PRMETR
PHTLIB,DL0:FORLIB
FILINI/O:1
MNTINI/O:1
FILCAL/O:1
//
SET SL OFF
UNLOAD SL
SET USR NOSWAP
RUN SONDSY
```

After this is done, you are in the new photometer system software ready to see if you have fixed the old bug and create any new ones. You can see that using these command files saves you an incredible amount of time and typing. The above process gives you enough time to go an refill your cup with coffee!

I have included one command file that compiles the program TSTCNT.FOR on the disk. While I wouldn't ordinarily leave lots of command files like this on the disk, I wanted to provide you with an example of compiling a program that is not related to the photometer system.

```
.@TSTCNT
FORT TSCNT
LINK TSTCNT, PHTCNT, PRMETR,DL0:FORLIB
```

Notice that there are some components from the photometer system that I am using in the routine TSTCNT, yet I do not want the whole library, or all the other aspects of the photometer system linked together here. I figured out which element I needed in the library (PHTCNT.MAC -- which controls the photometer counters), and then what other parts of the system I needed (PRMETR.MAC -- which contains the storage of all the global parameters) and of course the FORTRAN library.

A final command file is FPYCPY.COM which stands for "floppy copy" and is the backup for the photometer system. The floppy disk is called under RT-11, DY0:. Any writes and reads to and from this disk must use this label.

```
.@FPYCPY
copy dl2:*.sav  dy0:*.*      All the executable files
copy dl2:*.for  dy0:*.*      All the FORTRAN code
copy dl2:*.mac  dy0:*.*      All the assembly language code
copy dl2:*.com  dy0:*.*      All the command files
copy dl2:*.dat  dy0:*.*      All the data files (Make sure disk is clean)
squeeze dy0:
dir/print/full dy0:
```

This command file takes a while and copies all useful code over to a floppy disk for a backup. I recommend that you regularly backup your software as it gets modified. Once, all the files are copied, this software squeezes the disk and then sends a printout of the directory to the printer. This should then be stored with the floppy disk so you know the date of the backup. The more careful and systematic you are with backups, the more chance you'll have of keeping the system running in the field.

Presently there is no system bootable backup. In other words, if the system disk (DL0:) can't boot one day, then there is no other device to boot from. One of the first things you should do is to try and create a bootable floppy disk, and even maybe a bootable magnetic tape. You should read in the manuals about this (I didn't have time to do this for you, I'm sorry!) under COPY.

A first thing you could try is to mount a tape and initialize it and then type

```
.COPY/DEVICE DL0: MT0:
```

This should copy the system disk bit for bit onto the magnetic tape, and preserve the bootstrap file. However, I cannot guarantee that this will work.

Modification to hardware -- RADAR port

I added a parallel port to the system to read in the radar position information. This is a general 16 input port and can really be used for anything you want to interface to the computer, so hopefully it will be useful to have at other field locations as well.

The port is on the back of the Azimuth Elevation Controller and is a DB25 connector labelled RADAR INTERFACE. This goes to two 14 pin sockets on the interface board by ribbon cable connectors. This following is a list of the signal names and connections going to the DRV11-J interface in the PDP-11 computer (I enclosed documentation on this interface in the envelope for this controller).

LABEL	J2	DRV11-J	DIP	CABLE	DB25
AZ7 (MSB)	41	IN07	1	1	1
AZ6	36	IN06	14	2	2
AZ5	42	IN05	2	3	3
AZ4	35	IN04	13	4	4
AZ3	40	IN03	3	5	5
AZ2	38	IN02	12	6	6
AZ1	39	IN01	4	7	7
AZ0 (LSB)	37	IN00	11	8	8
GND	GND		5	9	9
GND	GND		10	10	10
EL7 (MSB)	45	IN15	6	11	14
EL6	46	IN14	9	12	15
EL5	43	IN13	7	13	16
EL4	49	IN12	8	14	17
EL3	48	IN11	1	15	18
EL2	44	IN10	14	16	19
EL1	50	IN09	2	17	20
EL0	47	IN08	13	18	21
GND	GND		3	19	22
GND	GND		12	20	23
GND	31	USER RDY D			

This radar port is read in the MACRO routine RDAZEL residing in file MNSTAT.MAC and stores the lower and upper bytes of this word as the Azimuth and Elevation of the Radar. Presently there are no checks to see if the data is valid or not. Since the Radar position can change at any time, there is the chance that a read could occur during this change and thus the data point could be strange. I didn't have time to address this problem, but your could modify the read to read the word twice and do a compare, and if it is not the same, then the data point is not valid -- try again. However, it might not be a big problem. Just keep an eye out for it in your data records.

SLAVING TO THE RADAR

The question of slaving to the radar was simply a question of not having the time to program the steps. There is, however, a very simple way to try and implement this. The real way to have one robotics axes slave to another is by using a feedback loop that is constantly re-adjusting the velocity of the axes trying to follow. This would require a lot of code, and while it is not conceptually hard, it would be time consuming to implement.

You might want to try a quick scheme using already existing software that could very easily be implemented. The basic idea would be to come up with a slewing rate that would be a reasonable one to follow the radar around during it's data taking runs. This would have to arrived at by trial and error, but utilizing the already existing command MNTSLW, you could just keep the radar and the mount somewhat near each other by issuing short MNTSLW commands every second, or so.

For instance, if the radar was at position Xr,Yr, and the mount was a position Xm,Ym, a MNTSLW command could be issued to move the mount to the present position of the radar:

CALL MNTSLW(Xr,Yr,SLVRTE)

where SLVRTE is the experimentally determined optimized rate to slave to the radar. The problem with this is that between every MNTSLW vector, the mount would stop, causing a kind of jerky motion. By the time this vector was done, the radar would be located at point X'r,Y'r, and the mount would be at Xr,Yr. Immediately, another short vector would be initiated:

CALL MNSTLW(X'r,Y'r,SLVRTE)

You could get even more extravagant by analyzing the distance and modifying SLVRTE to make the motion smoother (i.e. if really close, SLVRTE = RTEMIN, if medium distance, SLVRTE=RTEMED, and if really far away (i.e. when first entering mode) SLVRTE=RTEMAX).

This is not the ideal way to control a slaved device as the motion is somewhat jerky, but it has some good points to it. The first is that the telescope has a 5 degrees field of view verses the radar's one degree or less field of view. Thus, there should be some overlap. Also, during data acquisition of the radar scan, the radar is not moving that fast, so short slow motions of the mount should not be that much of a problem. From the programming point of view, this is extremely easy to implement. Presently, when the mount does it's scan, the photometer system just loops until it sees that the last move was completed (i.e. MNTSLW finished). Then it issues the next command (scan the opposite way). This could be changed so that when operating in SLAVE mode, instead of just slewing the other direction, it slews to the last read position of the radar (during the last data record dump). So, in routine SCANNR (located in file SONDSY.FOR):

```
      CALL PRINT ('SCANNR called')

      IF (SMODE .EQ.1) GOTO 1000              !Mode was Scanning

C  Mode is to SLAVE to the radar, check present positions and issue next motion command
C  First, check to see if radar is more than 1 degree away from mount

      DIFFAZ = IFIX((RADRAZ*1.4) - (AZANGL/20.))
      DIFFEL = IFIX((RADREL*1.4) - (ELANGL/20.))

      IF (DIFFEL.LE.1  .AND.  DIFFAZ.LE.1) RETURN        !No motion needed

C  Select different rates for Slaving

      DIFFAZ = IMAX(DIFFAZ,DIFFEL)            !Find major axis
      IF (DIFFAZ .LT. 5) SLVRTE = RTEMIN
      IF (DIFFAZ .GT.5  .AND. DIFFAZ.LT. 20) SLVRTE = RTEMED
      IF (DIFFAZ .GT.20) SLVRTE = RTEMAX

C  Issue slew command

      CALL MNTSLW ((IFIX(RADRAZ*1.4)),((IFIX(RADREL*1.4)),SLVRTE)
      RETURN

C  Mode is SCANNING, see which way mount is moving and issue next slew command

1000  IF (ELSTAT .EQ. 'ST') GO TO 10

.....etc.
```

This software example is very conceptual. There are lots of different ways that you could implement this. Your scheme could be more complicated to allow continuous motion by modifying MNTSLW which already handles accelerations and decelerations. However, keep in mind that at the same time, the filter is interrupting\\\. The more math the computer has to do during this period (especially floating point), the more likely it is that the computer might crash, that interrupts will not operate at constant rates and that the data recording will not be consistent. After each software modification, check that all these things are still working well.

This example is meant purely as a conceptual skeleton. Thought should be given to how you want to handle how often this gets called. Maybe it should only be called every time a data record is written, instead of every time a vector is completed as in the SCANNING mode. (This would reduce the Math overhead in comparing the mount position with the radar position. There are thousands of ways to implement this, however, the best way will be the simplest, the one with the least math, and one that is called infrequently.

Obviously it will take a great deal of trial and error to get this working and to get the slew rates matched up well, but if you have time up in Greenland, I think it would be a lot of fun, and it's really the best place to write such software, as you have the radar right there to test.

DATA Format of Photometer Counters

There could be a lot of confusion when analyzing the photometer counter data if you do not understand how the data is acquired so I will give a brief description at this point. The data is stored in a long word (2 words, 4 bytes, 32 bits) This is to allow a large dynamic range. Electronically, the data can only be read in a 8 bytes/read, so each byte is clock into the appropriate register in the computer memory during the RDCNTR command (see Counter documentation and PHTCNT.MAC software comments).

The thing to be careful of is that the data is in integer format, but all FORTRAN routines assume integers are only 16 bits long and the last bit is a sign bit in 2's complement format. Thus, if you look at an INTEGER*4 word (such as our data) and print it out, you are really only looking at the first 16 bits and even that's in 2's complement. To get around this is the software (see for example FILCAL.FOR, READTP.FOR, TSTCNT.FOR), we convert this value into a floating point number before we do anything with it (such as displaying data , doing arithmetic, etc.)

HIST(i,j) = AJFLT(ITEMP)

ITEMP was an INTEGER*4 data point used to store the result of a RDCNTR subroutine. This data point is then converted into a floating point number from 32 bit integer format. (example is from FILCAL.FOR) This is extremely crucial when analyzing your data or using it. If this is done, you have an incredible amount of dynamic range available and can utilize very long integration times if you need them.

Timing of DATA Records

I just wanted to mention that it seems when the software is dumping a data record to tape or disk, the interrupt rate is disrupted a little bit (perhaps 200 to 600 usecs). This probably is not that important, but it should be noted, and maybe really tested to see if it is true and what kind of effects it may have (i.e. period peaks in your data, when the integration time is a tiny bit longer). There may be ways around this as well.

Quick Reference to Using Photometer System

Assuming you understand the basic operation of the Photometers, and that you can figure out this software by the prompts and trial and error, and that you really don't want to read through this manual, the basic steps are:

1. Turn on the controllers and power supplies
2. Turn on the disk drives, and terminal
3. Turn on the computer
4. If computer didn't boot, hit the boot button once
5. Enter Date and Time when the computer prompts you for it.
6. Initialize the Magnetic tape (which needs to mounted with write ring!)
   @INIMT
7. Enter photometer system software
   @PH
8. You're all set!